НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

.

Дипломна робота

на тему: Розробка додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365

Студент групи ТМ-51 Битик Микола Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи професор кафедри АПЕПС,
д.т.н., проф. Ланде Дмитро Володимирович
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Кількість сторінок _____

Кількість ілюстрацій _____

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки”

на тему: Розробка додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365

Виконала: студента 4 курсу, групи ТМ-51

Битик Микола Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор кафедри АПЕПС,
д.т.н., проф. Ланде Дмитро Володимирович
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент старший науковий співробітник ІПРІ НАН України,
к.т.н., с.н.с. Каденко Сергій Володимирович
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студента

Битик Микола Олександрович
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365

керівник роботи старший науковий співробітник ІПРІ НАН України, к.т.н., с.н.с. Каденко Сергій Володимирович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи дані з серверів платформи Microsoft Office 365 Outlook
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення для ефективного менеджменту виконання доручень на підприємстві, описати програмну реалізацію програмного застосунку, розробити програмне забезпечення, розробити інтерфейс користувача

_____
_____
_____

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень)
1. Постановка задачі ефективного управління інформаційними потоками електронної пошти 2. Сучасний стан розвитку систем управління інформаційними потоками 3. Засоби розробки 4. Опис програмної реалізації 5. Робота користувача з системою 6. Висновки

Дата видачі завдання "10" жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	14.10.2018-23.12.2018	
2.	Розробка архітектури та загальної структури системи	2.02.2019-3.03.2019	
3.	Розробка структур окремих підсистем	4.03.2019-14.04.2019	
4.	Підготовка матеріалів	15.04.2019-18.04.2019	
5.	Програмна реалізація системи	18.04.2019-16.05.2019	
6.	Захист програмного продукту	17.05.2019	
7.	Оформлення пояснювальної записки	18.05.2019-3.06.2019	
8.	Передзахист	31.05.2019	
9.	Захист	17.06.2019-22.06.2019	

Студентка _____ Битик М. О .
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ланде Д.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота присвячена розробці додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365. Програмний продукт являє собою застосунок, що доповнює стандартний функціонал платформи Microsoft Office 365. Створений у вигляді веб-застосунку з використанням мов програмування Javascript, HTML, CSS та об’єднаний з Outlook за допомогою Azure Active Directory.

Система може використовуватися користувачами організацій, що використовують можливості платформи Office 365. Потенційними користувачами програмної системи є керівники та виконавці завдань на підприємстві.

Ключові слова: електронна пошта, система контрою виконання доручень, microsoft office 365, Azure Active Directory.

ABSTRACT

The thesis is devoted to the development of the application for managing information flow outlets on the platform office 365. The software product is an application that complements the standard functionality of the Microsoft Office 365 platform. It is created as a web application using Javascript, HTML, CSS and Connected to Outlook using Azure Active Directory.

The system can be used by users of organizations that use the capabilities of the Office 365 platform. Potential users of the software system are executives and executors of tasks at the enterprise.

Keywords: e-mail, mission control system, microsoft office 365, Azure Active Directory.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД База даних

SQL Structured query language —моваструктурованих запитів

СУБД Система керування базами даних

HTML Hypertext Markup Language — Мова розміткигіпертекстових документів

DOM Об'єктна модель документа

IDE Integrated development environment —Інтегроване середовище розробки

VCS, СКВ Version Control System— Системаконтролю версій

UI User Interface —Інтерфейс користувача

ПК Персональний комп'ютер

ОС Операційна система

JSON JavaScript Object Notation

ВСТУП

Робота в будь-яких організаціях вимагає узгодженої роботи усіх її членів. Проблеми організації взаємодії співробітників під час взаємної роботи зазвичай вирішуються використанням електронної пошти.

Як невід'ємний компонент якісної комунікації в організації є електронна пошта, надаючи можливість масових розсилань, передачі файлів та доручень для зручної взаємодії членів організації. Більшість електронних скриньок не надають керівникові достатньої інформації, щодо своєчасності реагування на отримані документи, вказівки, накази та доручення і таким чином зменшують своєчасність виконання доручень та не забезпечують достатньої комунікації членів організації

Наразі існують багато поштових скриньок такі як: Gmail, Ukr.net Yandex та Outlook. Кожний з них має функціонал по організації взаємодії між користувачами. Але, тільки Outlook тісно пов'язаний з Microsoft Office 365, що спеціалізований саме під потреби організації. В цій системі є можливість створити організації підключити до ней всіх її членів, та має гнучкі налаштування для контролю користувачів. А також можливість підключати свою додатки, що дає змогу ще більше розширити функціонал.

На сьогоднішній день Microsoft Office 365 є досить популярною в організацій різного розміру. Електронна пошта Outlook, яка є частиною онлайн офісного додатку Microsoft Office 365, що надає доступ членам організацій до всієї інформації, що зберігаються на хмарному сервері, та користуватися платформою будь де та будь коли при наявності інтернету. Але наразі в системі відсутня можливість контролю виконання доручень та моніторингу активності користувачів та функціоналу своєчасного реагування на зволікання зі сторони членів організації.

Для вирішення цієї проблеми пропонується розробити додаток до електронної пошти Outlook у вигляді веб-застосунку, що доповнюватиме функціонал та стане повноцінною частиною Outlook за допомогою хмарної служби управління посвідченнями і доступом від корпорації Майкрософт (Azure Active Directory).

1 ПОСТАНОВКА ЗАДАЧІ ЕФЕКТИВНОГО УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ ПОТОКАМИ ЕЛЕКТРОННОЇ ПОШТИ

Підчас проходження практики була поставлена задача реалізувати додаток для управління інформаційними потоками електронної пошти outlook на базі платформи office 365. У вигляді додатку до Outlook, який стане частиною інтерфейсу та буде надавати додаткові функції для роботи з системою. Для ефективного контролю членів організації та аналіз виконавчої дисципліни відповідно до доручень.

Розроблена система повинна забезпечувати наступні можливості:

— розсилка листів,
— масова розсилка,
— розсилка запитів на зведення,
— надання доручень,
— контроль ознайомлення з документами,
— аналіз виконання запитів на зведення,
— повторно відправити листів при необхідності,
— аналіз виконавчої дисципліни відповідно до доручень (наявності та своєчасності відповідей).

Для розробки додатку використати будь-які мови програмування, використовуючи сучасні програмні технології.

2 СУЧАСНИЙ СТАН РОЗВИТКУ СИСТЕМ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ ПОТОКАМИ ЕЛЕКТРОННОЇ ПОШТИ OUTLOOK НА БАЗІ ПЛАТФОРМИ OFFICE 365

Електрона пошта стала не від'ємною частиною нашого повсякденного життя. Не дивно, що наразі практично всі компанії і організації користуються електронною поштою, вона зручна для обміну листами, контрактними пропозиціями, інформацією та комерційною інформацією. Відсутність у підприємства електронної пошти з погляду людей, яким відома її зручність і практичність, істотний мінус.

Всесвітнім стандартом для комунікації в організаціях рахується електронна пошта. Компанія Microsoft в цьому добилася гарних успіхів створивши систему Office 365, яка орієнтована на бізнес, там малі організації. Платформа Office 365 має велику кількість функцій, що є корисними для більшості організацій. Функції що дозволяють працювати з файлами Word, OneNote, PowerPoint і Excel онлайн за допомогою браузера, або мобільного застосунку. Також є можливість редагувати один файл декільком користувачам одночасно у браузері. Функціонал Microsoft Planner дозволяє робочій групі легко створювати плани, призначати завдання, обмінюватися файлами, спілкуватися на робочі теми та зручно перевіряти виконання доручень всіх членів групи.[1]

Платформа Office 365 має внутрішній застосунок для комунікації в швидкому темпі, який зроблено у форматі Messager, за допомогою якого можливо створювати чати робочих груп та мати можливість передавати аудіо- та фото повідомлення.

3 ЗАСОБИ РОЗРОБКИ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій. Середовищем розробки інформаційної системи було обрано JetBrains WebStorm. Для створення графічного інтерфейсу системи використовувався HTML, CSS.

Для розробки алгоритмів використовувалась мова програмування JavaScript.

Для розробки сервера використовувалась мова програмування Node.js.

Також використовувалися сучасні фреймворки Angular, AngularJS, AdarJS.

3.1 Середовище розробки WebStorm 2017

Середовище розробки WebStorm один з застосунків компанії JetBrains, який було обрано для розробки додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365.

Середовище розробки JetBrains WebStorm — середовище розробки, що дозволяє працювати з мовами програмуваня JavaScript, CSS & HTML, розроблена на основі платформи IntelliJ IDEA та схожа з нею по логікі роботи з застосунком. Має схожий інтерфейс та велику кількість однакових функцій (рисунок 3.17.) [8].

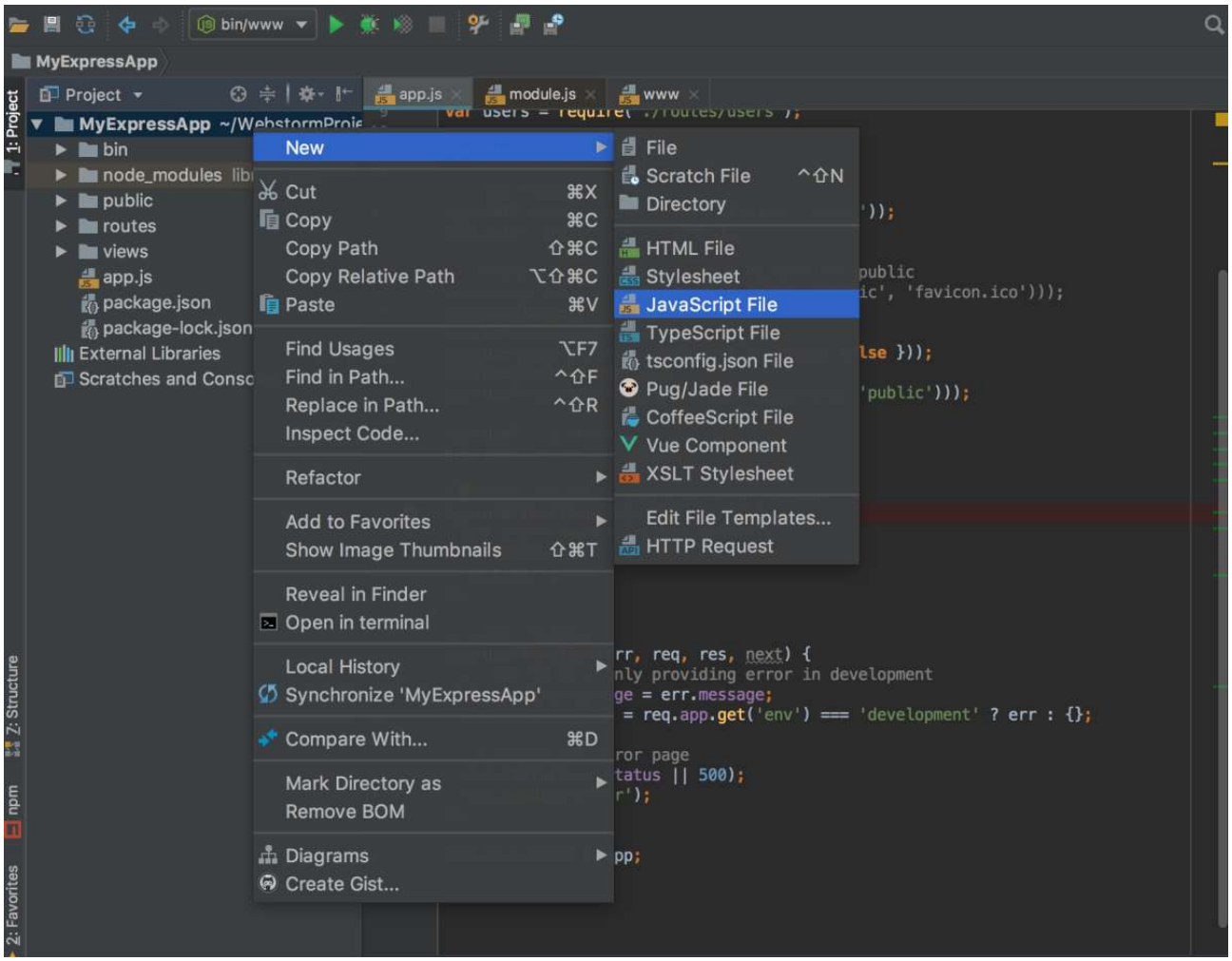

Рисунок 3.1 — Інтерфейс середовища розробки JetBrains WebStorm

Основні плюси роботи з WebStorm:

— інтелектуальний редактор з підтримкою синтаксису, пошуку в документації та рефакторингу для мов та середовищ: JavaScript, Node.js, ECMAScript 6, TypeScript, CoffeeScript і Dart, а також для HTML, CSS, Less, Sass і Stylus;

— аналіз написаного коду, вказує на помилки та дає можливість швидко їх виправити;

— можливість підключати до роботи сучасні фреймворки;

— зручна навігація по проекту і рефакторинг;

— вбудований відладчик для клієнтського коду і Node.js;

— інтеграція з системою контролю версій Github, та багато іншого.

Середовище розробки JetBrains WebStorm надає можливість підключати до роботи сучасні фреймворки, такі як: React, Vue.js, Angular і Meteor, а також розробки на стороні сервера з Node.js. і Cordova і Ionic для мобільної розробки.

Відладчик WebStorm має функціонал аналізу проектів, щоб доповнювати його при необхідності. Допомога в кодуванні є значною перевагою WebStorm над іншими середовищами розробки.

Помилки в граматиці написання коду підкреслюються відразу після його написання, що дає змогу відразу звертати на помилки увагу та виправляти їх. Функція аналізу якості коду в усіх файлах проекту дає змогу виявити глобальні помилки та поліпшувати свої навички.

Платформа IDE дозволяє налаштувати інтерфейс під себе, а саме стиль коду для будь-якої мови, переноси строк, контроль відступів, фон, та стиль інтерфейсу і т. д. Власний стиль проекту можна зберігати у файлі, та при необхідності ділитися з іншими розробниками або використовувати в своїх цілях при зміні робочої станції.

Середовище розробки JetBrains WebStorm надає розширений функціонал для роботи з браущерами. Він вбудований прямо в IDE, тому відпадає необхідність перемикатися між редактором і браузером для роботи над застосунком.

Відладчик WebStorm має шари областей видимості, включаючи глобальні, локальні змінні та фрейми. Значення змінних відображаються поруч в тому ж рядку в редакторі. Це дозволяє перевіряти значення складних виразів JavaScript під час роботи з кодом. Точки зупинки підтримують режим призупинення та умов для зупинки та полегшують роботи з кодом.

Середовище розробки JetBrains WebStorm має можливість компілювати код TypeScript в JavaScript за допомогою вбудованого компілятора. Параметри компіляції можливо вводити вручну або за допомогою JSON-файлів. Всі помилки в роботі компілятора відображаються в редакторі в реальному часі.

Середовище розробки JetBrains WebStorm має уніфікований інтерфейс для роботи з системами керування версіями, що дає можливість працювати в git, SVN, Mercurial і Perforce в інтуїтивно зрозумілому інтерфейсі.

Всі незафіксовані зміни в проекти з системи контролю версій виділяються у редакторі і у вікні проекту. Існує можливість легкої відміни будь-яку зміни всього за два кліка. Інструмент злиття файлів швидко і інтуїтивно вирішує всі конфлікти.

.

3.2 Мова розмітки HTML

Термін HTML (HiperText Markup Language) з’явився не так давно, а саме після створення інтернету. Знадобилася універсальна мова створення інтернет сторінок, яка б була легкою в розумінні, але в той самий час задовольняти всі потреби тогочасних програмістів та бізнес, що переходив в епоху інтернету. Головним атрибутом мови розмітки HTML є гіперпосилання. Теги та їх структура чітко обумовлена для полегшення читання коду, що дозволяє підтримувати на покращувати реалізовані проекти.

При загрузці веб-сторінки на екран браузера проводиться відповідно вказівками прописаних в HTML файлі. Проте велика кількість авторів не в змозі описати в файлі всіх деталей, якщо не описані деталі не є критично важливими. Тоді програма, що видає документ, використовує успішно запам’ятовані нею — при створенні або в процесі налаштування користувачем — значення. Через це, одна і та ж HTML-сторінка може відображатися по різному в різних браузерах та машинтах.

По замовчуванню в HTML-сторінці не описуються шрифт текстової частини документу — не вказано ні розмір шрифту, ні назви шрифту. Отож, якщо видача на машині з визначеним за замовчуванням значенням розміру шрифту в 14 пунктів буде сильно відрізнятися, ніж на машині зі шрифтом в 18 пунктів. Навіть при однакових шрифтах на екранах різного розміру сторінка буде виглядати інашке.

Основними елементами мови HTML по своїй суті є теги. Теги прописуються в дужках. Відкрита дужка визначає початок тега з атрибутами такими як: id, class, та інші. Що дають змогу відрізняти теги для роботи з ними, змінювати окремі елементи сторінки. Закрита — відзначає кінець цього тега. Все, що знаходиться між дужками є

однаковими з точки зору цих властивостей. Також є можливість вкладати теги один в один.

3.3 Формати зберігання та обміну даних

Для зберігання та обміну даними розроблявся мова з простим формальним синтаксисом, зручний для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті. Мова називається розширюваним, оскільки ним не фіксується розмітку, яка використовується в документах: розробник вільний створити розмітку відповідно до потреб конкретної області, будучи обмеженим лише синтаксичними правилами мови.

Для обміну даними між різними платформами та системами використовуються текстовий формат. Прикладами таких форматів є XML, JSON, CSV.

У розробленій застосунку, формат JSON було використано для зберігання тимчасових даних, які потрібні для роботи застосунку. Цей формат досить розповсюджений та більшість мов програмування мають вбудований інтерпретатор формату JSON.

3.4 Формат JSON

Назва JSON розшифровується як JavaScript Object Notation. В порівняні з XML JSON лаконічніший та бути більш придатним для створення складних структур (рисунок 3.2.) [9].

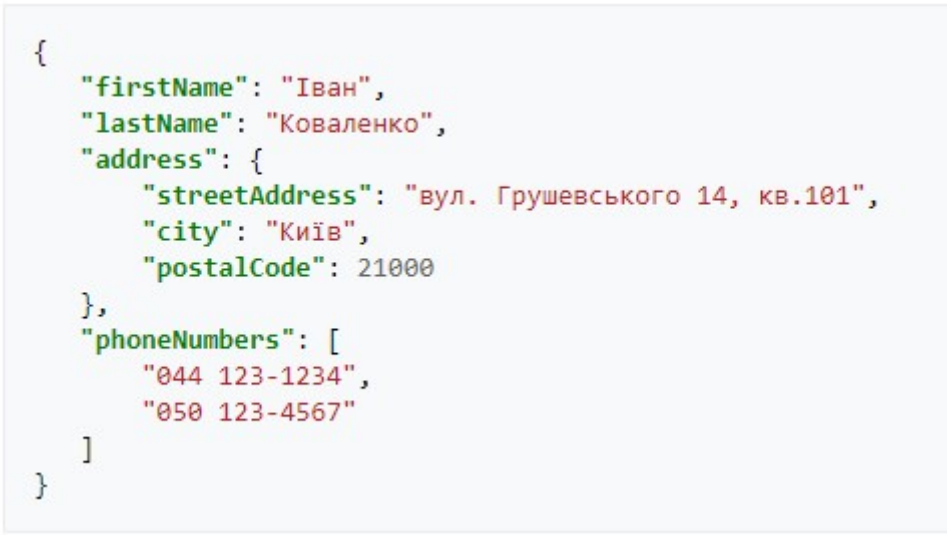

Рисунок 3.2 – Приклад структури формату JSON

Формат JSON – це текстовий формат, який повністю незалежний від мови, але використовує угоди, знайомі програмістам сімейства мов C, включаючи C, C++, C#, Java, JavaScript, Perl, Python і багато інших. Завдяки цим властивостям цьому JSON є ідеальною мовою обміну даними.

В JSON будується на двох типах структур:

— Упорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

— Колекція пар ім'я/значення. У різних мовах це реалізується як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативний масив.

Дані у форматі JSON (RFC 4627) являють собою:

— масиви [..];

— JavaScript-об'єкти { ... };

— або значення одного з типів:

- рядок;
- число;
- логічне значення thue/fals;

- null.

Об'єкт – це невпорядкований набір пар ім'я/значення. Об'єкт починається з "{" закінчується "}" . За кожним ім'ям слідує: (двокрапка), а пари ім'я / значення поділяються комами (рисунок 3.3.) [10].

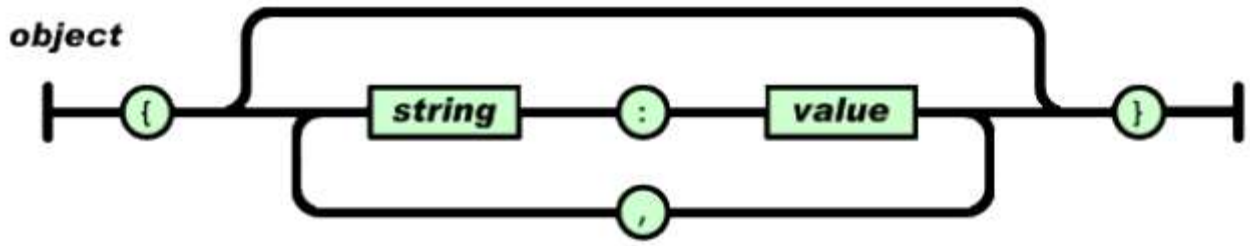

Рисунок 3.3 – Візуалізація об'єкту в JSON

Формат JSON містить 3 пари значень, що виглядає так: "key": "value", "key": "value"," key":"value", кожна пара значень розділена комою. Пари ключових значень мають двокрапки між собою, як наприклад тут "key" : "value" [10].

Зазвичай при роботі з даним створюють файл формату JSON для відправки через Інтернет або локальну мережу. У більшості випадків для опису структури і типів даних використовується спеціальний файл JSON-схеми, за допомогою якого формується документація, що пояснює формат даних та надає приклади використання.

Перевагами використання JSON-схеми є:

— можливість генерації документації на основі схеми;

— можливість перевірки даних, на основі схеми, що є корисним для автоматизованого тестування;

— опис вже існуючої структури даних.

JSON-схема створюється у форматі JSON, приклад зображено на рисунку 3.4.

Рисунок 3.4 – Структура JSON схеми.

Використовують декілька спеціальних властивостей, що є обов'язковими при визначенні схеми:

Для управління версіями програмних інтерфейсів (API) використовують ключове слово "$schema", що вказує на те, що ця схема написана відповідно до конкретного стандарту.

Ключове слово "$id" визначає URI для схеми і базовий URI, з яким дозволяються інші посилання URI в схемі.

Для опису даних використовують ключові слова анотації "title" та "description". Вони є корисними для формування документації, хоча и не визначають обмежень до отриманих даних.

Для обмеження типу даних, що можуть міститися в документі JSON використовується ключове слово "type".

Властивість "properites" схеми повинно містити об'єкт зі списком полів, що описують структуру та тип даних JSON-документу. Кожне поле також може мати опис та бути необов'язковим, для цього використовують властивість "required" зі значенням false в описі поля.

3.6 Мова програмування TypeScript

Для створення єдиного класу-інтерфейсу для доступу до віддаленого провайдера та отримання даних і класів, що реалізують окремі структури даних інтерфейсами мови Typescript.

Мова програмування TypeScript базується на основі JavaScript. Популярність ідей нової мови призвела до того, що ряд з цих ідей в подальшому стануть актуальною і цей день частиною нового стандарту JavaScript. А нова версія одного з досить популярного фреймворків для Web — Angular повністю написана на TypeScript завдяки співпраці програмістів з компаній Microsoft і Google [13].

Розробником мови є Андерс Гейлсберг, що створив такі мови програмування як Delphi, C# та Turbo Pascal [14].

Мова TypeScript забезпечує безпеку типів під час компіляції для коду JavaScript. Так як мова є зворотньо сумісним з не типізованою мовою JavaScript типи є повністю необов'язковими. Після компіляції програми є можливість виконувати код в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js. TypeScript створили як надмножина JavaScript з додатковою перевіркою типів. Завдяки цьому також дозволяє використовувати усі бібліотеки написані мовою JavaScript. Також, залишити існуючі JavaScript-проекти можна в незмінному вигляді, а у вигляді анотацій в окремих файлах розмістити дані про типізації, які не заважатимуть прямому використанню проекту [16].

Однак, для написання коду, вам необхідно знати JavaScript так як TypeScript просто стандартизує всі можливі способи надання хорошої документації по проектам на JavaScript.

На відміну від інших мовами, що компілюються в Javascript, Typescript:

— не надає новий синтаксис мови для існуючих можливостей Javascript, адже це не допомагає виправити помилки;

— не є мовою, що суттєво відрізняється від мови програмування Javascript, що дозволяє не абстрагуватися занадто далеко від середовищ виконання та спільнот розробників.

Завдяки мові TypeScript в наступному стандарті ES6 мови JavaScript (рисунок 3.5) планують додати ряд функцій, навіть для середовищ, які не підтримують ці функції для коду Javascript (старі веб-браузери та середовища виконання, такі як Node.js).

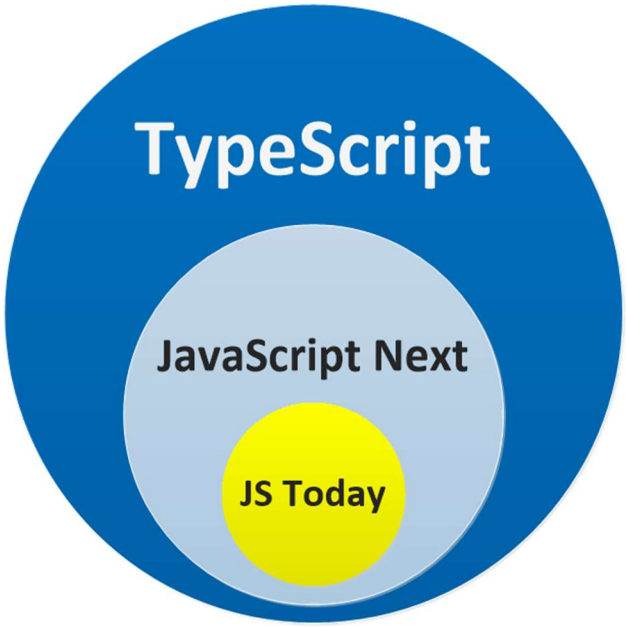

Рисунок 3.5 — Ієрархія мови Typescript

Так як Typescript є нетипізованою, то типи можуть бути визначені неявними — компілятор виведе якомога більше інформації про тип, щоб забезпечити розуміння типів з мінімальними витратами при розробці коду. Наприклад, у наступному коді TypeScript буде знати, що зміна foo має номер типу число, і покаже помилку при зміні типу рядок (рисунок 3.6.).

Оскільки мова TypeScript є Open Source, що значить що всі її інструменти доступні для всіх бажаючих. Для роботи з TypeScript можна використовувати будь-яку операційну систему. Отож в будь-якому текстовому редакторі можливе написання коду. Більшість середовищ розробки та деякі текстові реактори мають

підтримку TypeScript на рівні плагінів, що дозволяє скористатися низкою переваг цих застосунків, наприклад, можливість працювати в привичному інтерфейсі та використовувати всі гарячі клавіші, підсвічуванням синтаксису або спливаючими підказками для типів і конструкцій мови. [17].

```
var foo = 123;
foo = '456'; // Error: cannot assign `string` to `number`

// Is foo a number or a string?
```

Рисунок 3.6 — Приклад неявної типізації

Також є можливість визначити типи явно, за допомогою анотації типів.

Анотації використовуються для:

— визначення типом компілятору;

— для документації коду для наступного розробника, якому доведеться працювати з кодом;

— примусового визначення типу змінної.

Мова TypeScript використовує постфіксні анотації типів (рисунок 3.7), при роботі з явно визначеними типами.

```
var foo: number = 123;

var foo: number = '123'; // Error: cannot assign a `string` to a `number`
```

Рисунок 3.7 — Приклад явної типізації в TypeScript

Так як TypeScript є дуже схожою на JavaScript, в ній було додано підтримку структурних типів [18]. Для кращого розуміння розглянемо наступний приклад. Функція iTakePoint2D буде приймати додаткові параметри, навіть ті що не визначено

явно (рисунок 3.8).

```
interface Point2D {
    x: number;
    y: number;
}
interface Point3D {
    x: number;
    y: number;
    z: number;
}
var point2D: Point2D = { x: 0, y: 10 }
var point3D: Point3D = { x: 0, y: 10, z: 20 }
function iTakePoint2D(point: Point2D) { /* do something */ }

iTakePoint2D(point2D); // exact match okay
iTakePoint2D(point3D); // extra information okay
iTakePoint2D({ x: 0 }); // Error: missing information `y`
```

Рисунок 3.8 — Структурні типи даних в TypeScript

Приклад інтерфейсу зображено на рисунку 3.9.

```
1  interface ClockInterface {
2      currentTime: Date;
3      setTime(d: Date);
4  }
5
6  class Clock implements ClockInterface {
7      currentTime: Date;
8      setTime(d: Date) {
9          this.currentTime = d;
10     }
11     constructor(h: number, m: number) { }
12 }
```

Рисунок 3.9 — Використання інтерфейсів в TypeScript

У TypeScript також можливе одне з найбільш поширених застосувань інтерфейсів в таких мовах, як C # і Java, - явне забезпечення відповідності класу певному контракту

3.7 Мова програмування JavaSvript

Мова програмування JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування. Найчастіше всього використовується для створення сценаріїв для веб-сторінок, що надає можливість взаємодіяти з сервером на стороні клієнта [19] змінювати структуру та зовнішній вигляд веб-сторінки, асинхронно обмінюватися даними з сервером, керувати браузером.

Мову JavaScript називають прототипоною, що означає що вона є підмножиною об'єктно-орієнтованих мов, скриптову мову програмування з динамічною типізацією [20]. Також в мові JavaScript окрім прототипної частки є частина імперативного фунціоналу і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу [21].

3.8 Технологія Node.js

Технологія Node.js дозволяє створювати веб-сервери та мережеві інструменти, використовуючи JavaScript і набір "модулів", які обробляють різні функціональні можливості. Модулі передбачають введення / виводу файлової системи, мережеві (DNS, HTTP, TCP, TLS / SSL або UDP), двійкові дані (буфери), функції криптографії, потоки даних та інші основні функції. Модулі Node.js використовують API, призначений для зменшення складності написання серверних додатків[23].

Програми Node.js можуть працювати на серверах Linux, macOS, Microsoft Windows, NonStop та Unix. Крім того, вони можуть бути написані з CoffeeScript (альтернатива JavaScript), Dart або TypeScript (строго типізовані форми JavaScript) або будь-яка інша мова, яку можна скомпілювати в JavaScript[23].

Node.js використовується, перш за все, для побудови мережевих програм, таких як веб-сервери. Найбільша різниця між Node.js і PHP полягає в тому, що більшість функцій в блоці PHP до завершення (команди виконуються лише після завершення попередніх команд), а функції Node.js не блокуються (команди виконуються одночасно або навіть паралельно, і використовують зворотню відповідь завершення сигналу) [23].

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Для поліпшення роботи та комунікації в організації потрібна система, яка буде в себе включати всіх членів організації та всіх дійових осіб надавати зручний та зрозумілий функціонал для взаємодії між ними. Значно перевагою буде, якщо система міститиме в собі усі функції для автоматизації взаємодії між її членами. Microsoft Office 365 є найкращою хмарною системою для організації роботи в команді. Office 365 надає змогу розробляти застосунки для певних задач, функції яких не реалізовані в системі. Застосунок над Outlook э найліпшим на даний момент для підвищення управлінських функцій керівника.

Архітектура клієнт-сервер дозволяє організувати роботу з системою на будь-якому комп'ютері та мати доступ до системи при наявності інтернету. Уся важлива інформація зберігається на серверах Microsoft Office 365, що забезпечує безпеку даних. Клієнтам надається відповідний доступ до даних. Для роботи з системою розроблений веб-інтерфейс, який було інтегровано до Outlook, що дозволяє взаємодіяти з системою у зручному середовищі. Сервер було вирішено розробити автономним, що дозволяє запускати застосунок лише на внутрішніх системах, тобто для доступу до нього, не потрібно використовувати підключення до інтернету, а лише те, щоб комп'ютери були з'єднані в межах однієї мережі, а також даний підхід збільшує надійність системи, адже кількість каналів по котрим можна провести атаку зменшується у незлічиму кількість разів.

Для отримання інформації з серверів Microsoft Office 365 було проаналізовано базу даних Microsoft Office 365 та визначено з якими таблицями має взаємодіяти застосунок. Це дозволяє на пряму брати інформацію з бази даних, працювати з нею та збільшує швидкість пошуку даних, адже зменшує область пошуку .

Для роботи з системою в організації мають бути передбачені 3 ролі:

- Керівник
- Підлеглий
- Адміністратор

В кожній з ролей передбачений свій рівень доступу до системи і кожен має свій функціонал.

4.1 Ролі та функції

Для повноцінної роботи системи передбачені три ролі: керівник, підлеглий та адміністратор. Між ними розподілені весь основний функціонал. Для кращого розуміння будуть наведені UML діаграми. Use case діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання. Use case діаграми наведені на наступних рисунках: рисунок 4.1 – функції курівника, рисунок 4.2 – функції підлеглого, рисунок 4.3 – функції адміністратора.

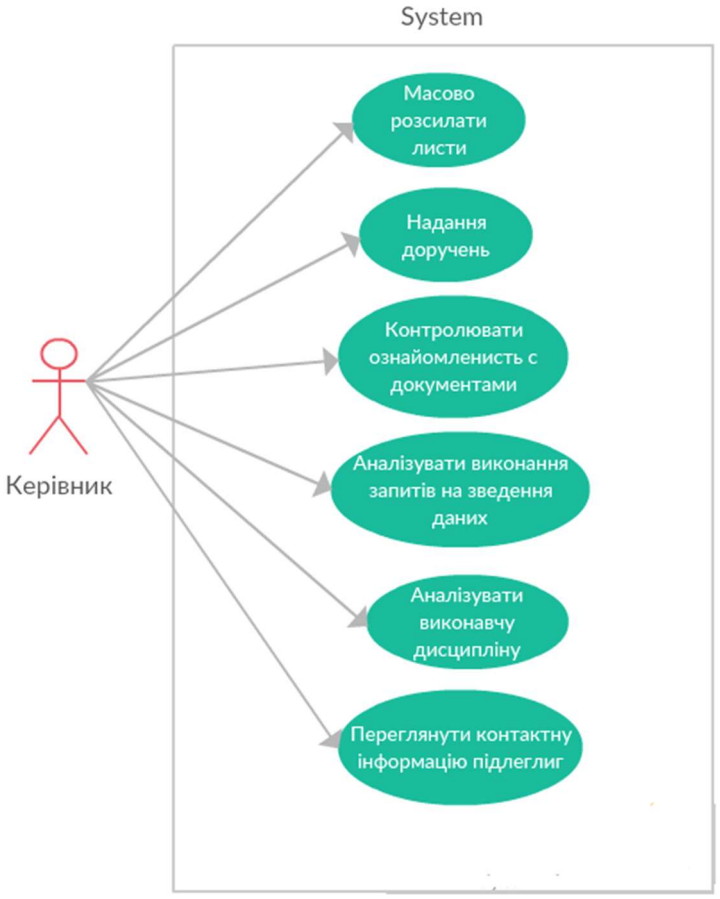

Рисунок 4.1 — Функції керівника

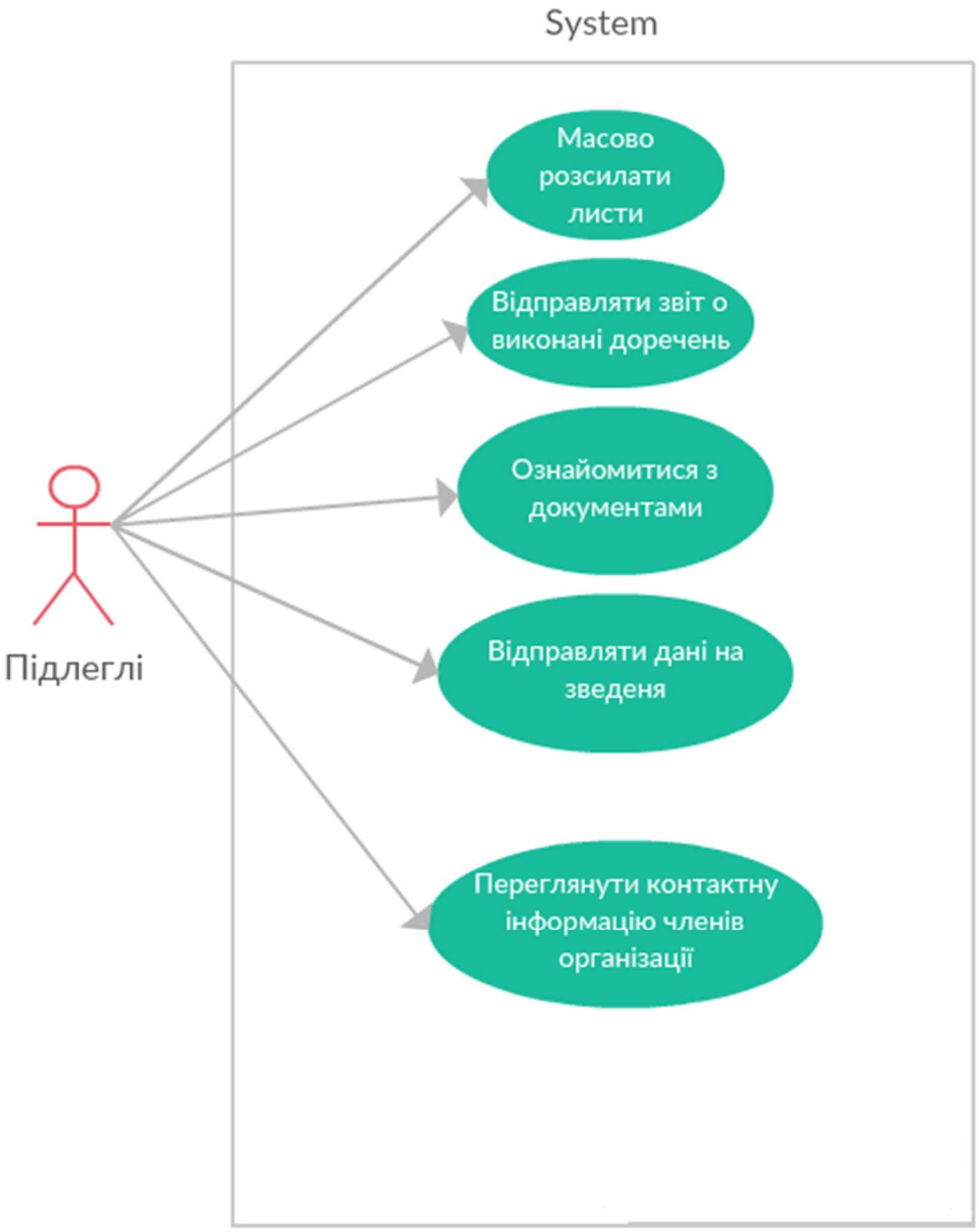

Рисунок 4.2 — Функції підлеглих

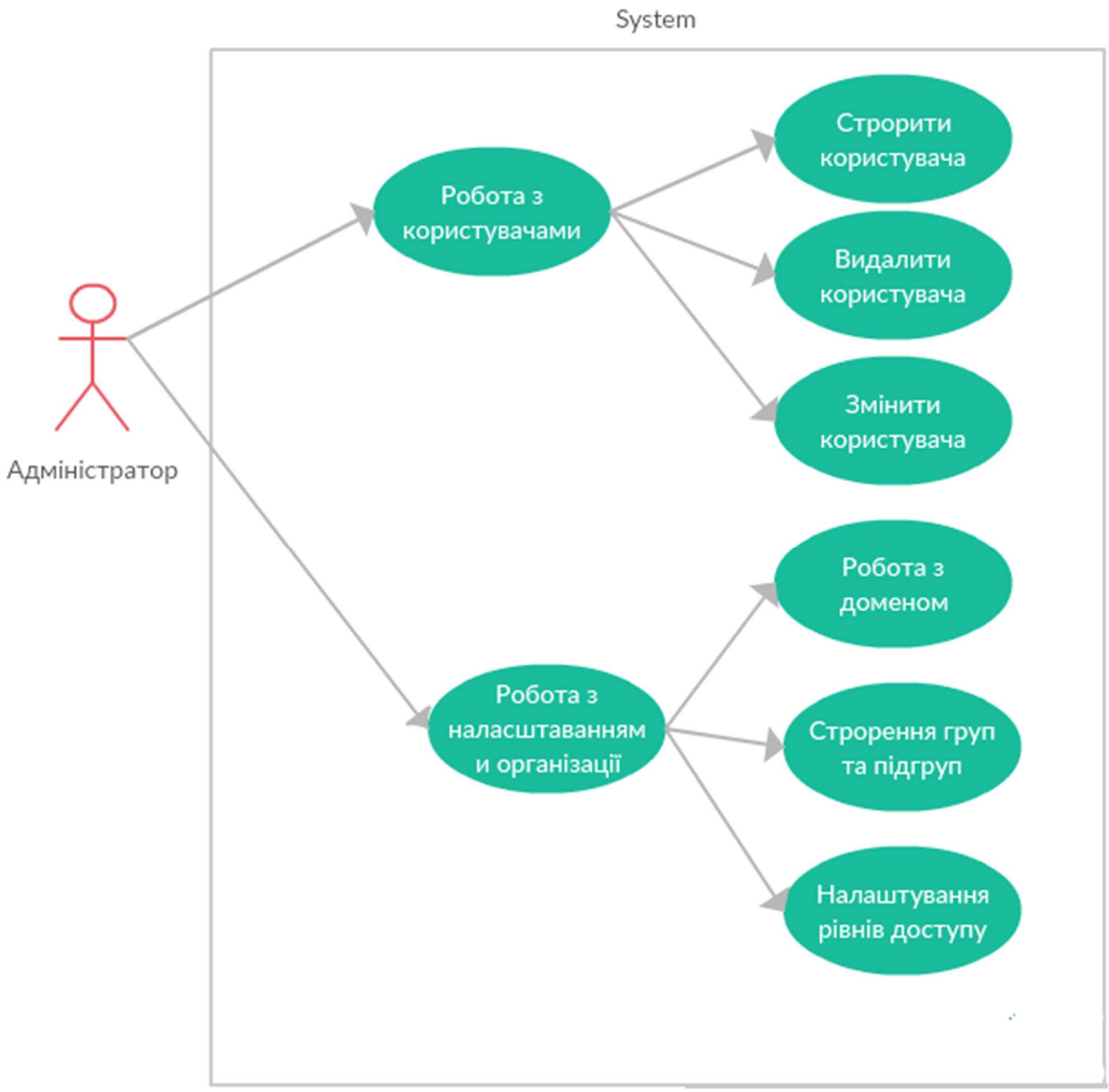

Рисунок 4.3 — Функції адміністратора

Use case діаграми дають повне розуміння для чого буде використовуватися застосунок та охарактеризує кожну роль на підприємстві. За допомогою UML діаграм стало зрозуміло, якою має біти архітектура проекту.

4.2 Архітектура програмного комплексу

Для створення застосунку було розроблено архітектуру програмного комплексу(рисунок 4.4). Це діаграма, як легка в розумінні, описує всі модулі програми та їх взаємодію одне з одним.

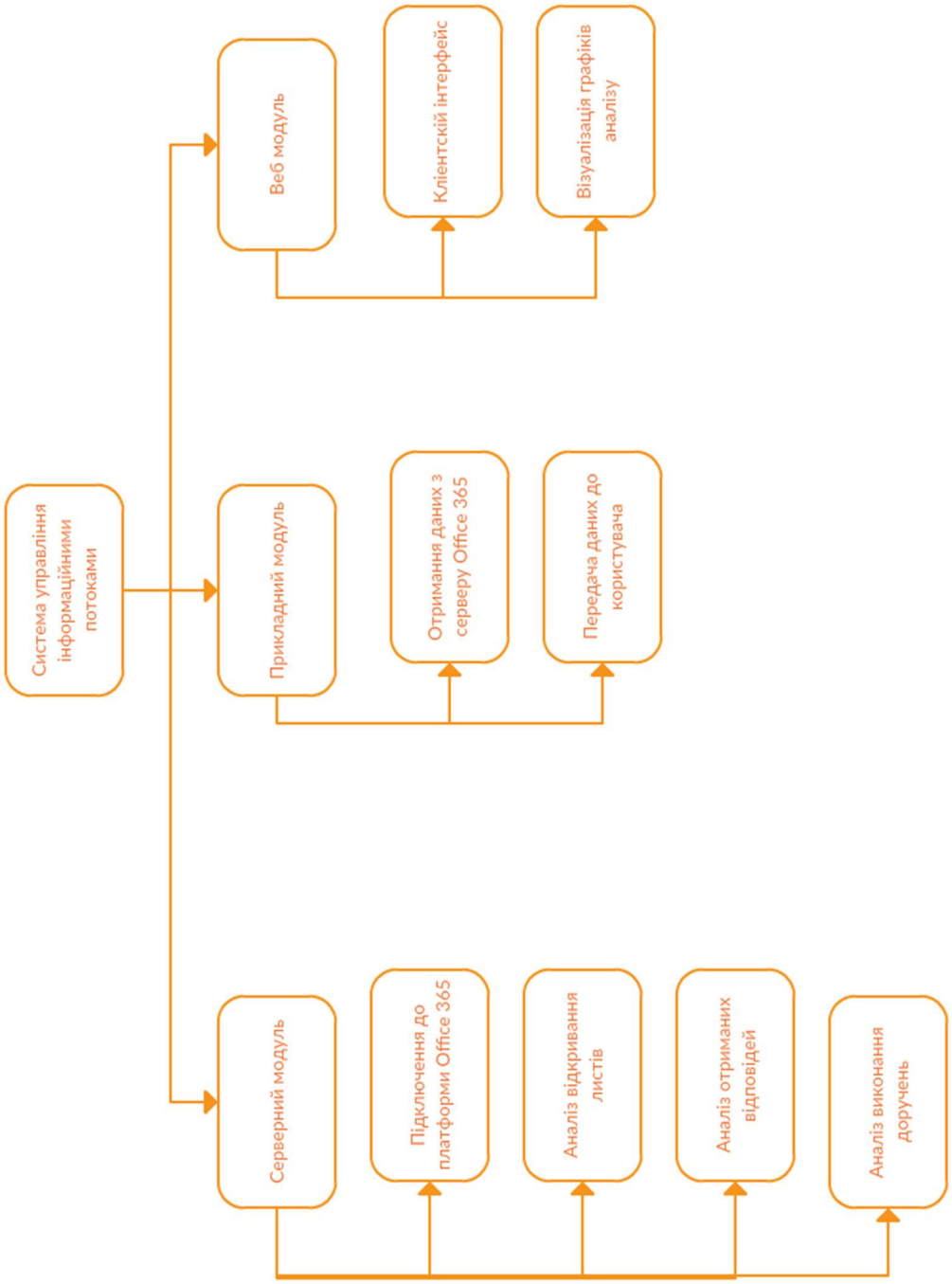

Рисунок 4.4 — Архітектура програмного комплексу

Створення архітектури програмного забезпечення — спосіб структурування програмної або обчислювальної системи, абстракція елементів системи на певній фазі її роботи. Система може складатись з кількох рівнів абстракції і мати багато фаз роботи, кожна з яких може мати окрему архітектуру.[24]

Дослідження архітектури програмного забезпечення допомагає визначити як найкраще розбити систему на частини, як ці частини визначають та взаємодіють одна з одною, як між ними передається інформація, як ці частини розвиваються поодинці і як все вищеописане найкраще записати використовуючи формальну чи неформальну нотацію[25].

4.3 Опис бази даних

Під час розробки архітектури програмного застосунку, ми продумали, що будемо звертатися до бази даних Microsoft Office 365, адже вся потрібна, для роботи застосунку, інформація зберігається на хмарних серверах Office 365. На рисунку 4.5 зображено таблиці бази даних Microsoft Office 365 до яких звертається розроблений застосунок.

В цих таблицях бази даних зберігається вся не обхідна інформація для реалізації функцій:

— розсилка листів,

— масова розсилка,

— розсилка запитів на зведення,

— надання доручень,

— контроль ознайомлення з документами,

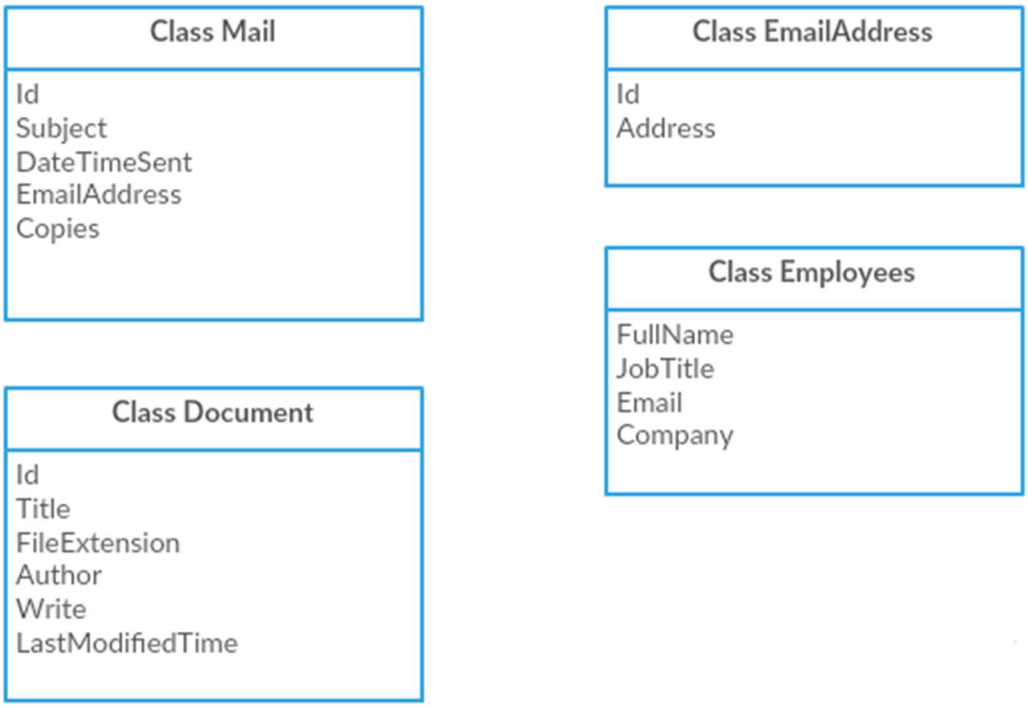

Рисунок 4.5 — Використання інтерфейсів

— аналіз виконання запитів на зведення,

— повторно відправити листів при необхідності,

— аналіз виконавчої дисципліни відповідно до доручень (наявності та своєчасності відповідей).

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

5.1 Системні вимоги

Встановлений пакет Node.js.

У разі потреби використання стосунку для розробки, при потребі у виправленні недоліків чи покращення функціоналу, потрібно встановити усі необхідні пакети для нормального функціонування застосунку. Це можна зробити з використанням команди npm install. Також в якості альтернативного інструменту для роботи з Javascript пакетами можна використовувати утиліту npm.

5.2 Інсталяція та налаштування програмного продукту

Для початку потрібно загрузити програмний додаток на свій персональний комп’ютер. В кореневий папці натиснути кнопку лівий «shift» та правою клавішою миші, у випадаючому меню вибрати «Відкрити вікно PowerShell тут» (рисунок 5.1).

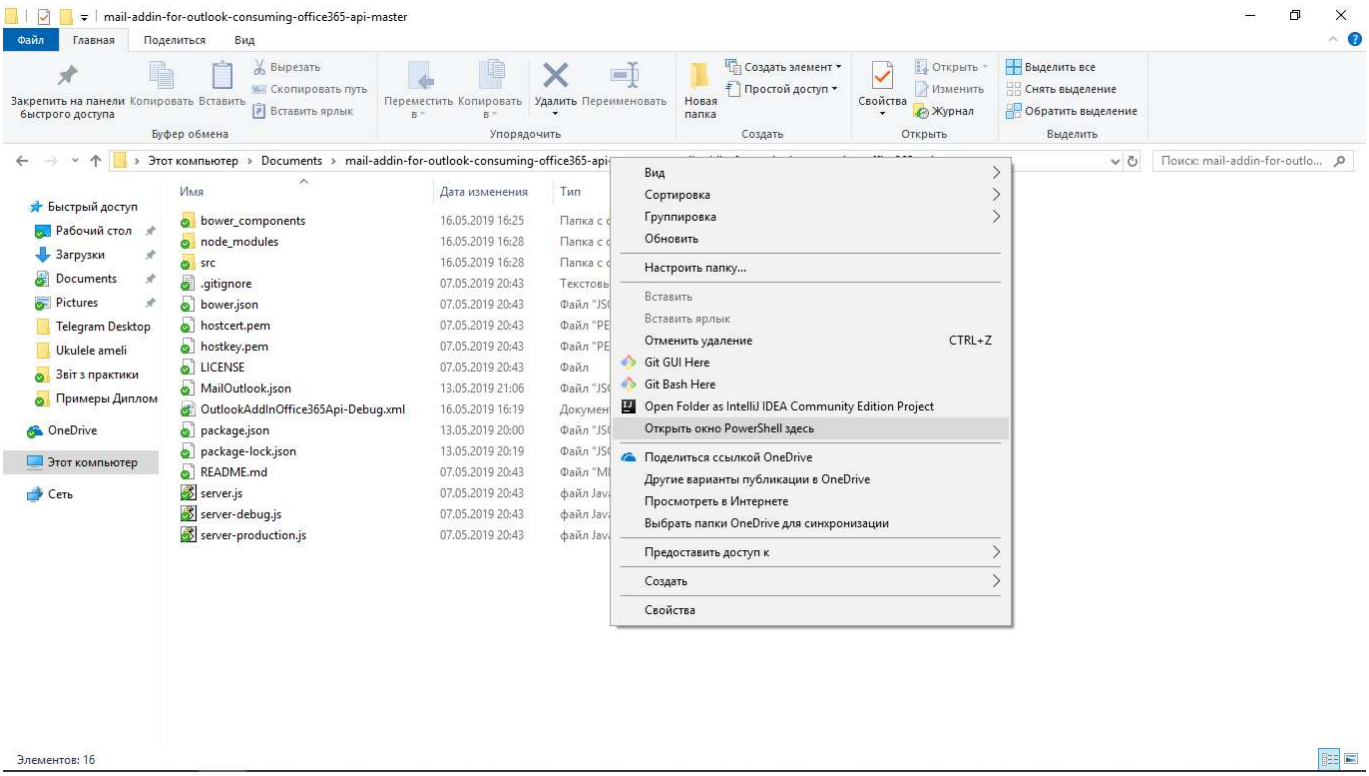

Рисунок 5.1 – відкрити вікно PowerShell

При першому запуску для коректної роботи всіх компонентів застосунку потрібно встановити усі необхідні пакети. Це потрібно зробити у два кроки, спочатку за допомогою команди npm install bower. Після завершення завантаження використати наступну команду npm install (рисунок 5.2).

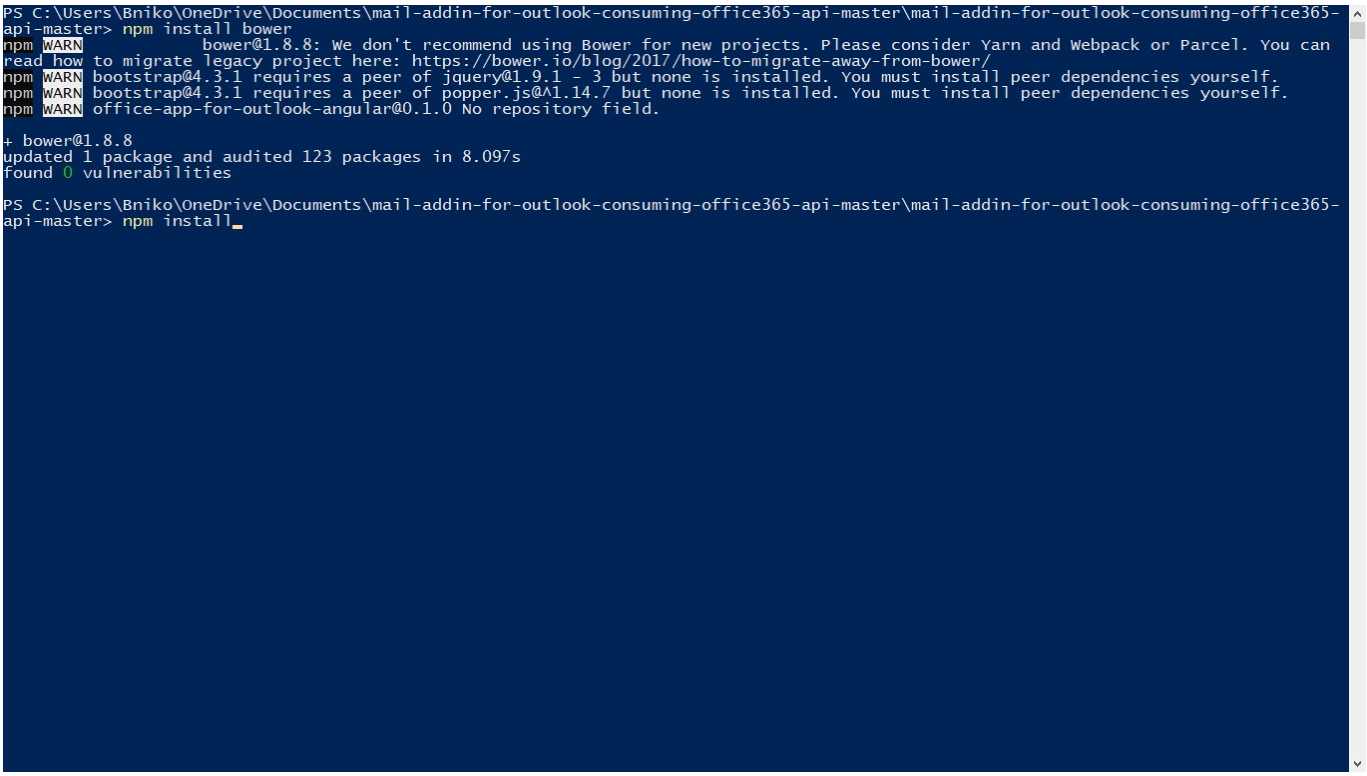

Рисунок 5.2 – вікно PowerShell з командами для інсталяції

Після завершення завантаження ми має змогу більше не повторювати попередні кроки при наступному запуску додатку.

Для подальшої роботи нам потрібно запустити сервер, це робиться теж с панелі PowerShell за допомогою команди npm start (рисунок 5.3). Для перевірки чи зробили все правильно відкрийте браузер та перейдіть «https: // localhost: 8443 / # /»

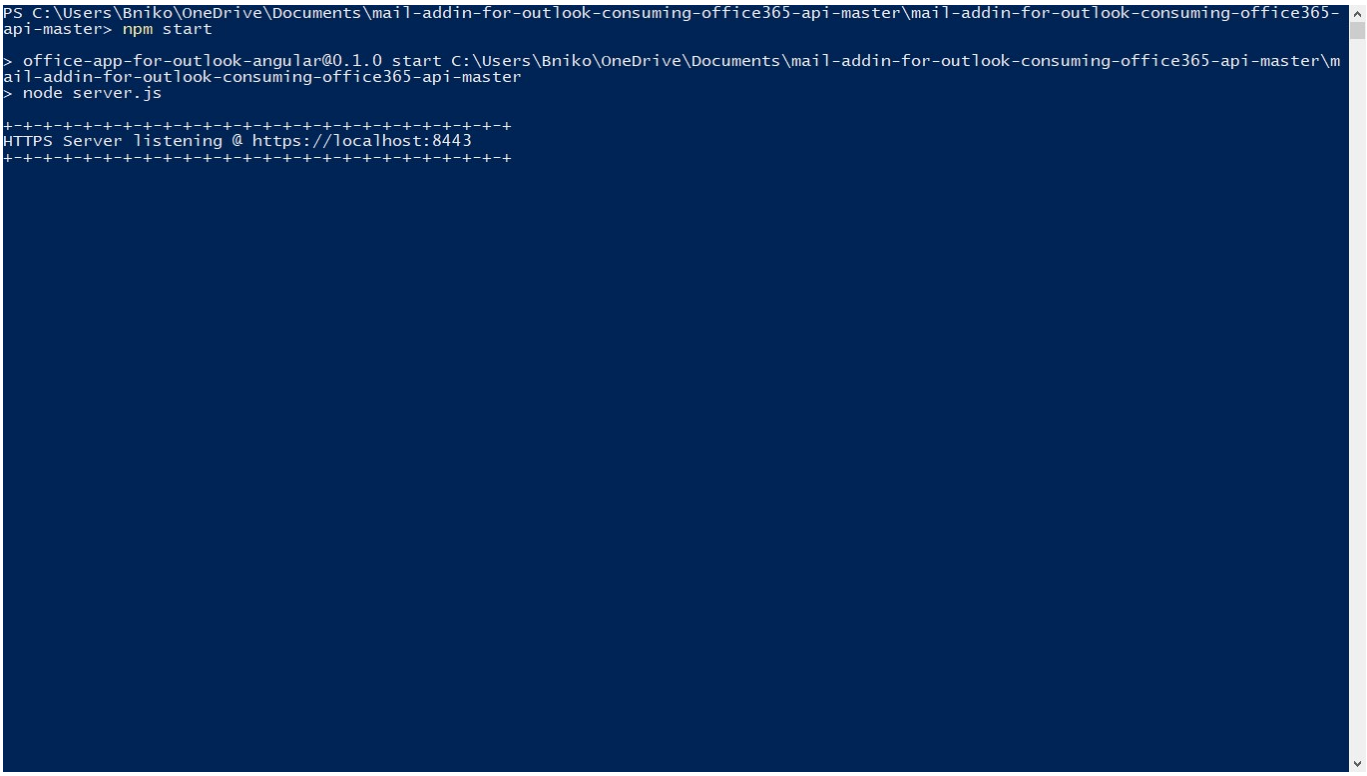

Рисунок 5.3 – запуск северу за допомогою PowerShell

Щоб мати змогу користуватися додатком ми маємо підключити його Office 365. Зробити це можливо за допомогою Azure Active Directory (рисунок 5.4). Потрібно зареєструвати свій додаток на сайті «https://portal.azure.com». У лівому меню вибрати «Azure Active Directory», потім «App Registration(Legacy)». Після цього ми маємо можливість зареєструвати новий додаток (рисунок 5.5).

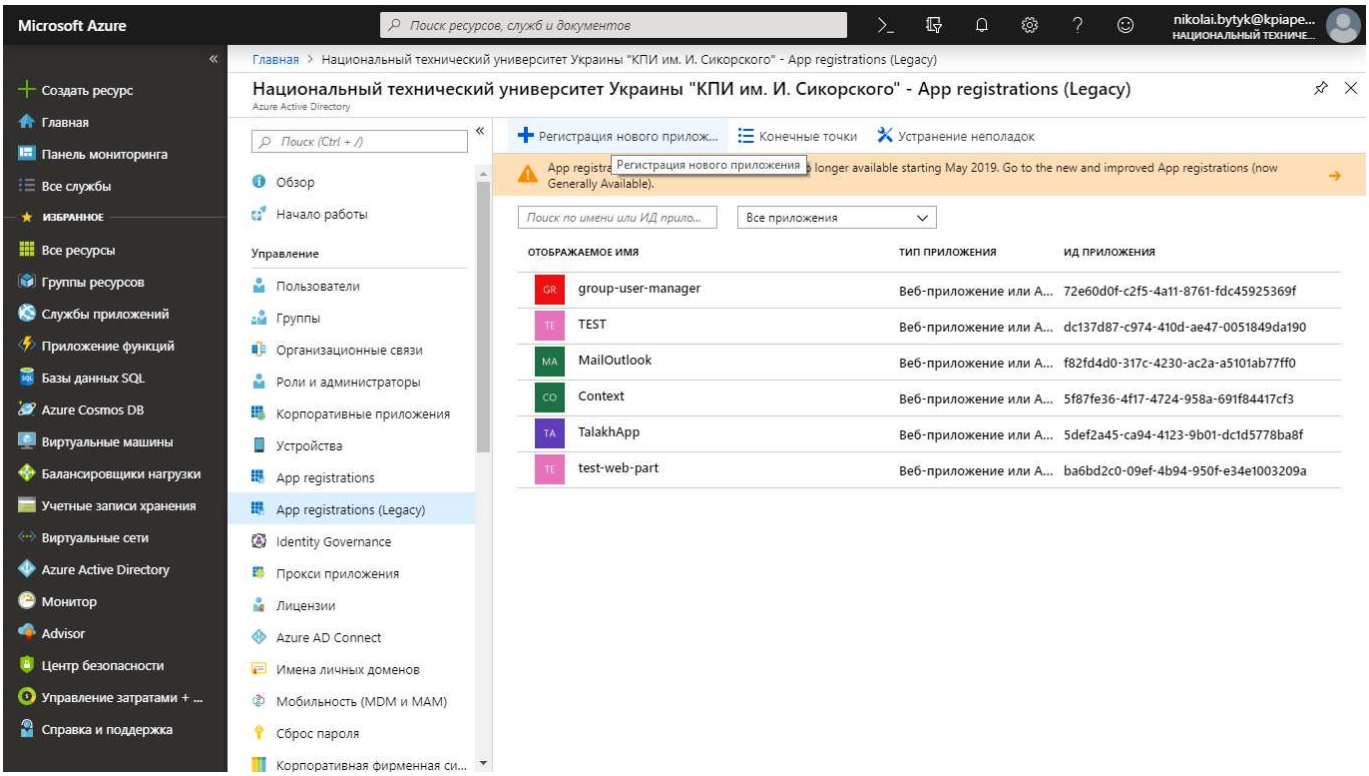

Рисунок 5.4 – azure active directory

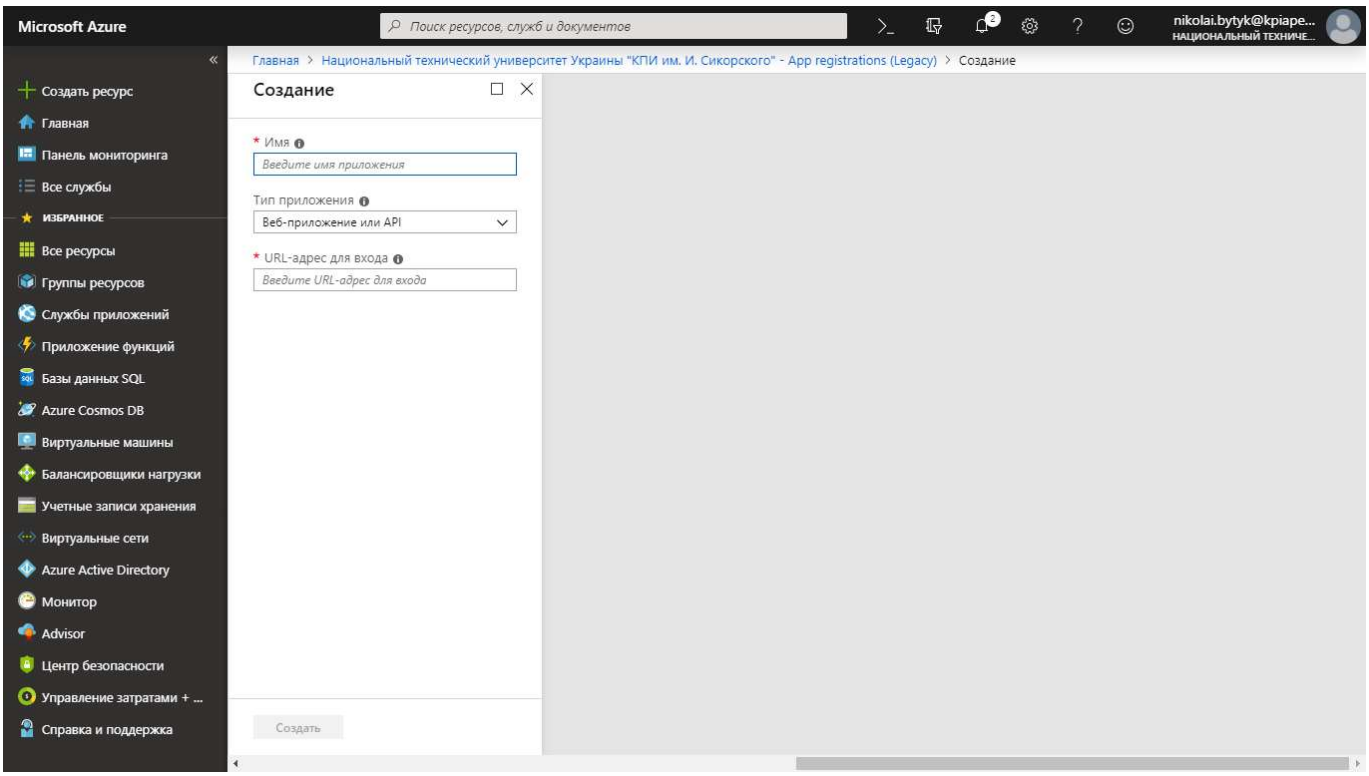

Рисунок 5.5 – App Registration(Legacy)

Після завершення реєстрації переходите на Outlook та дивіться на результат.

Виконавши всі описані вище кропи, ми маємо змогу користуватися застосунком.

5.3 Робота користувача з програмним продуктом

Інтерфейс електронної пошти Outlook має наступний вигляд (рисунок 5.6):

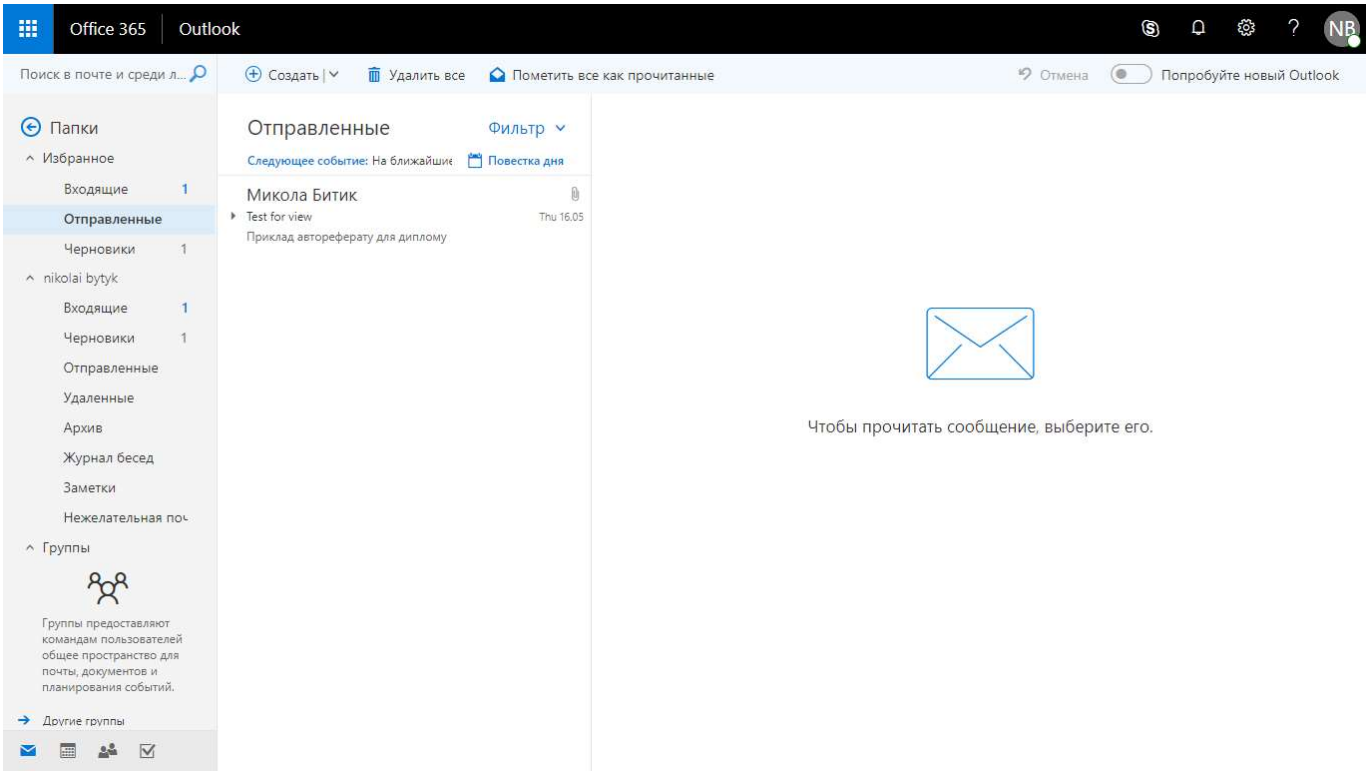

Рисунок 5.6 – Інтерфейс електронної пошти Outlook

Щоб система Microsoft Office 365 мала всі функції в організації має бути адміністратор, задача якого налаштувати систему під організацію. Для цього в системі передбачено Microsoft admin center (рисунок 5.7).

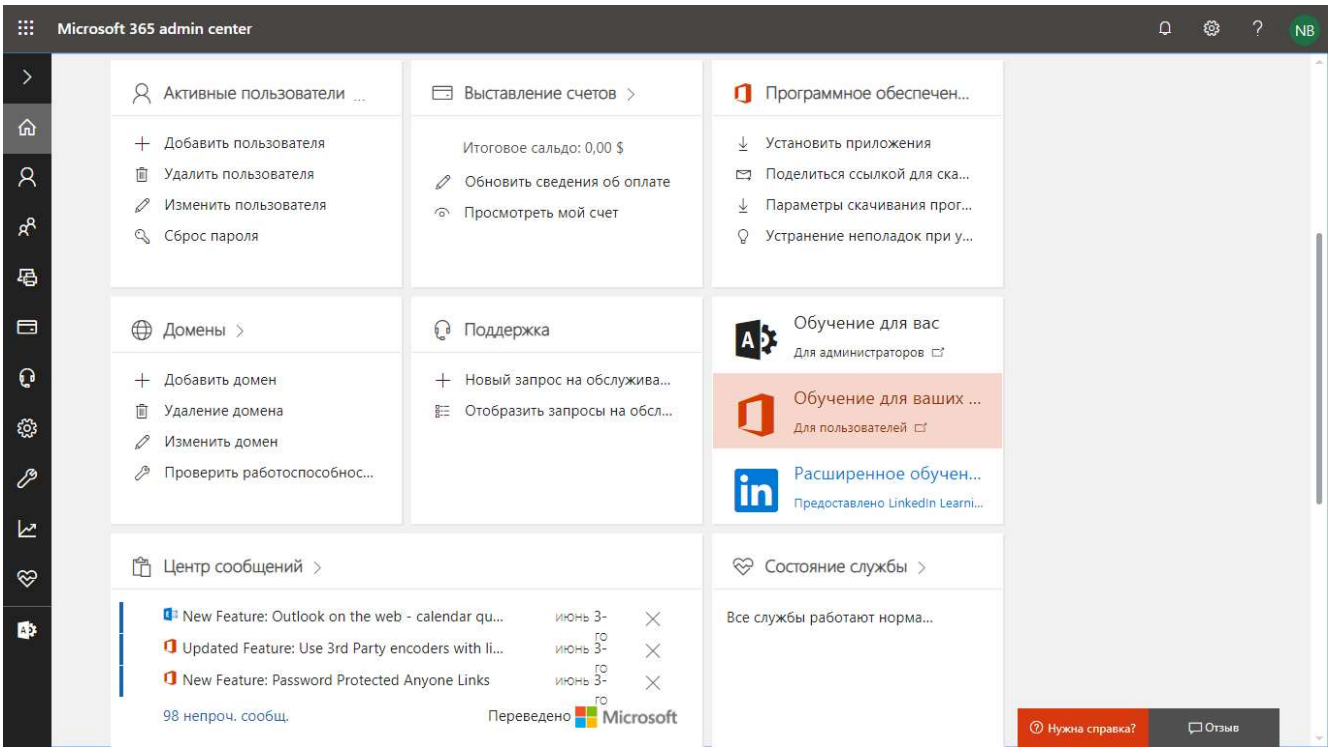

Рисунок 5.6 – Інтерфейс панелі адміністратора Microsoft admin center

Задача адміністратора створити в системі всій користувачів відповідно до членів організації. Розділити їх на відділи або групи, кожному користувачу надати відповідний доступ. Та завершити настройку системи.

Для роботи з додатком спочатку потрібно увійти у систему office 365 під своїм обліковим записом (рисунок 5.8). Увійшовши в систему, користувач бачить головну сторінку з інформацією про себе. Також він може перейти на головну сторінку, внести зміни до свого профілю Зайти на сайт Outlook, відкрити будь-якого листа та натиснути на кнопку «Context» (рисунок 5.9).

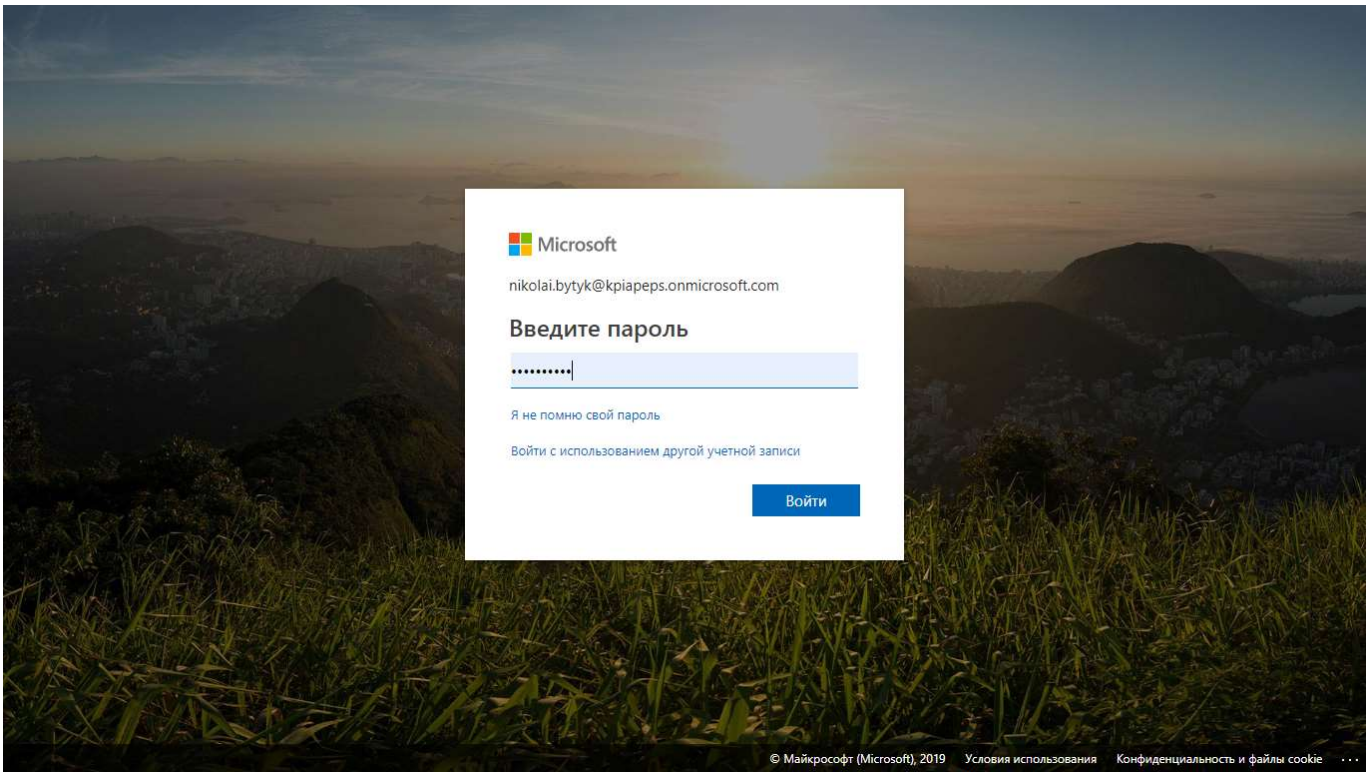

Рисунок 5.8 – Сторінка входу до Office 365

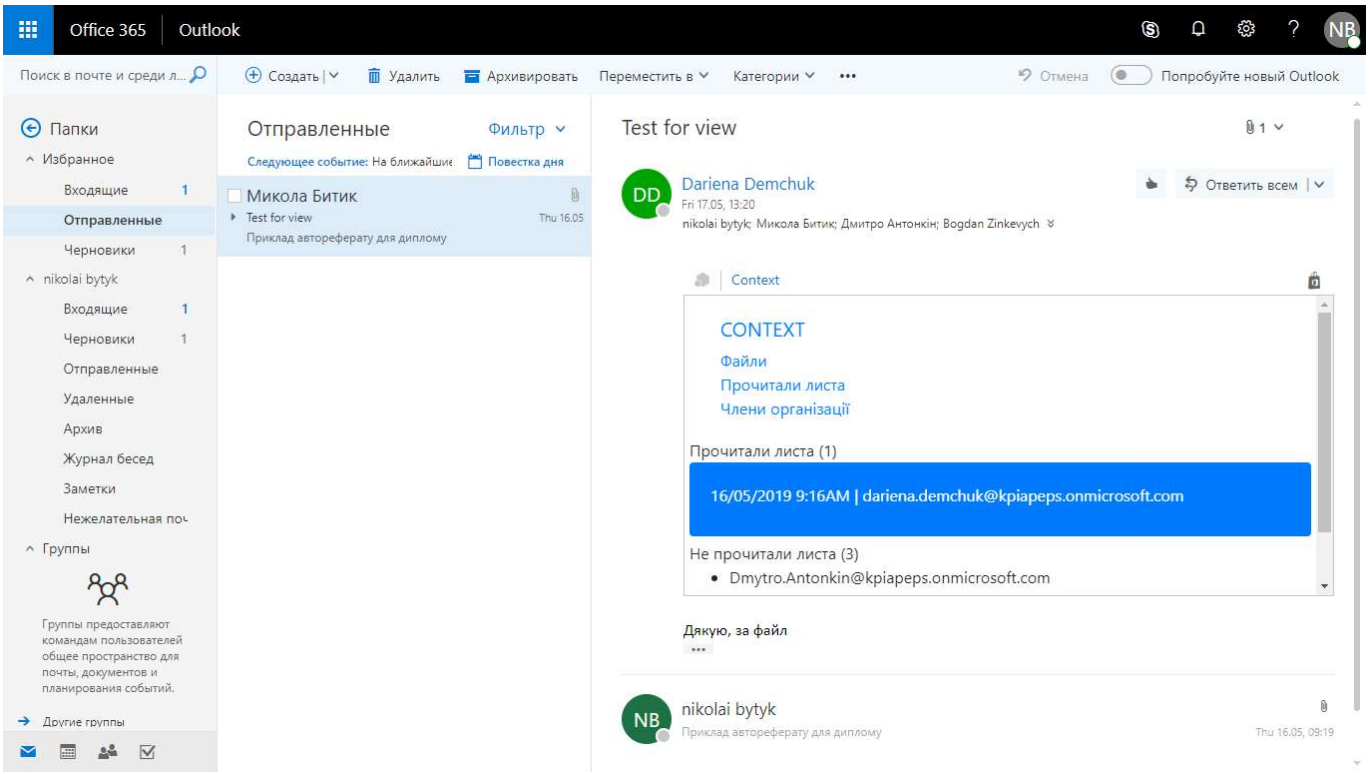

Рисунок 5.9 – Outlook з підключеним додатком

В відповідному вікні ви побачите 3 активні посилання: Файли, Прочитали листа, Члени організації.

В вкладці «Файли» на екрані відображається вся детальна інформація про всі прикріплені файли до листа (рисунок 5.10). Інформація така як: назва файлу, розширення файлу, інформацію про автора, дати створення та редагування.

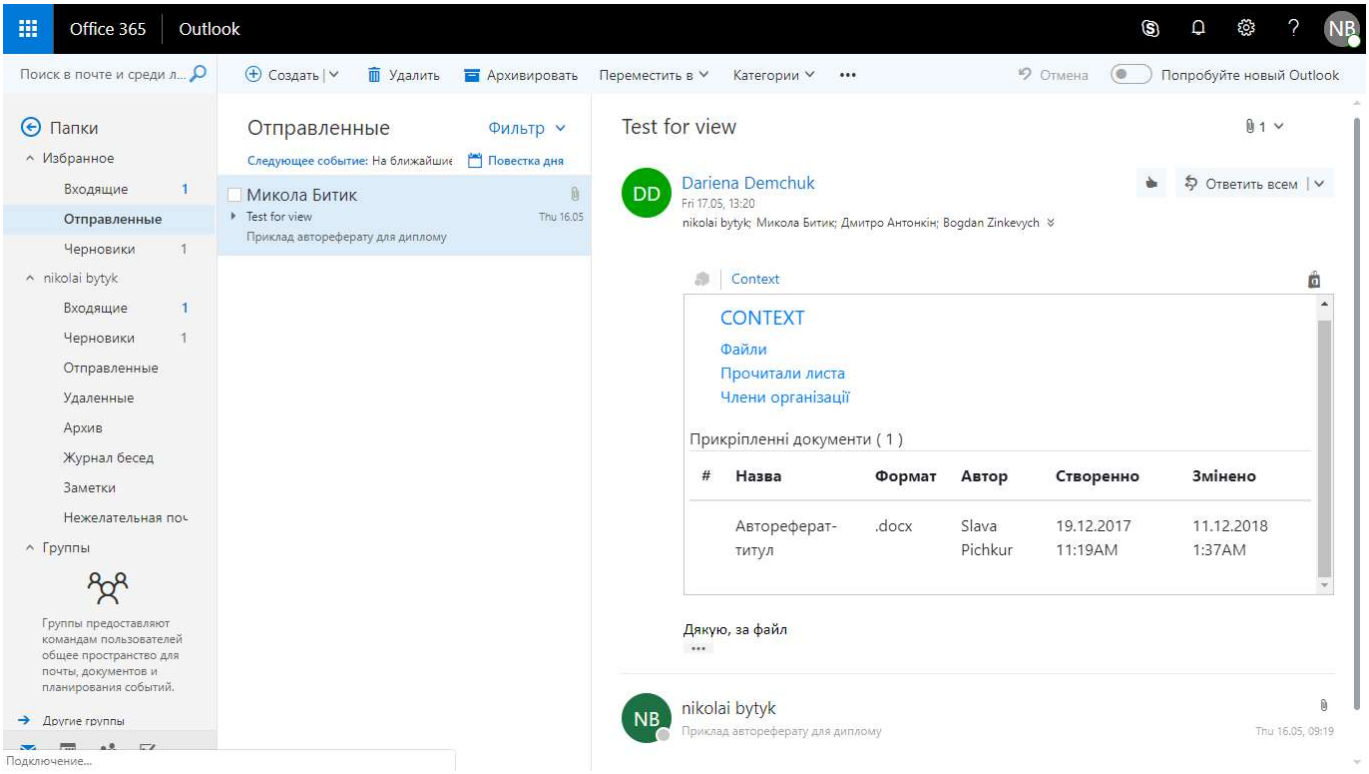

Рисунок 5.10 – інформація про прикріплені файли

В вкладці «Прочитали листа» на екрані відображається інформація про членів організації, які вже прочитали листа з детальною інформацією про дату та час та перелік користувачів, які не відкрили листа (рисунок 5.11).

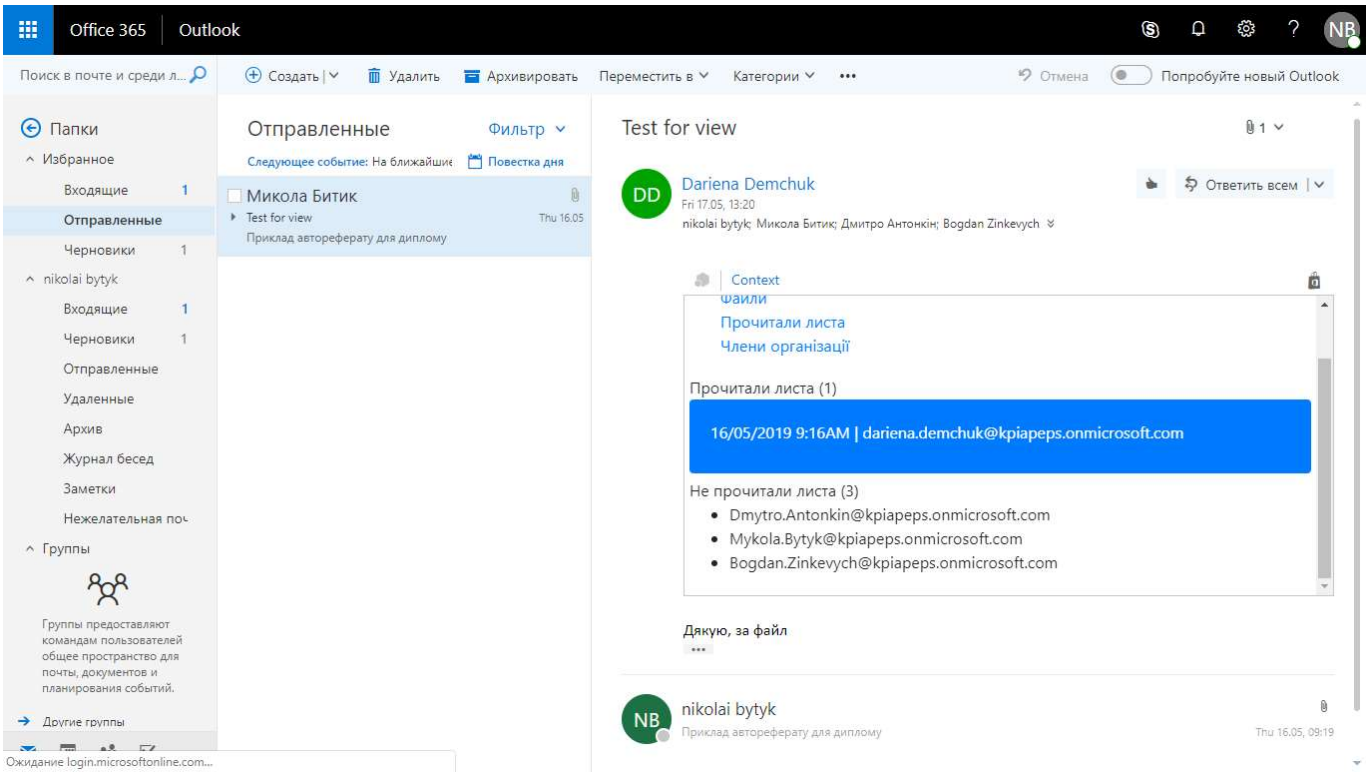

Рисунок 5.11 – активність членів організації

В вкладці «Члени організації» на екрані відображається інформація про членів організації, які включені в цю масову розсилку (рисунок 5.12).

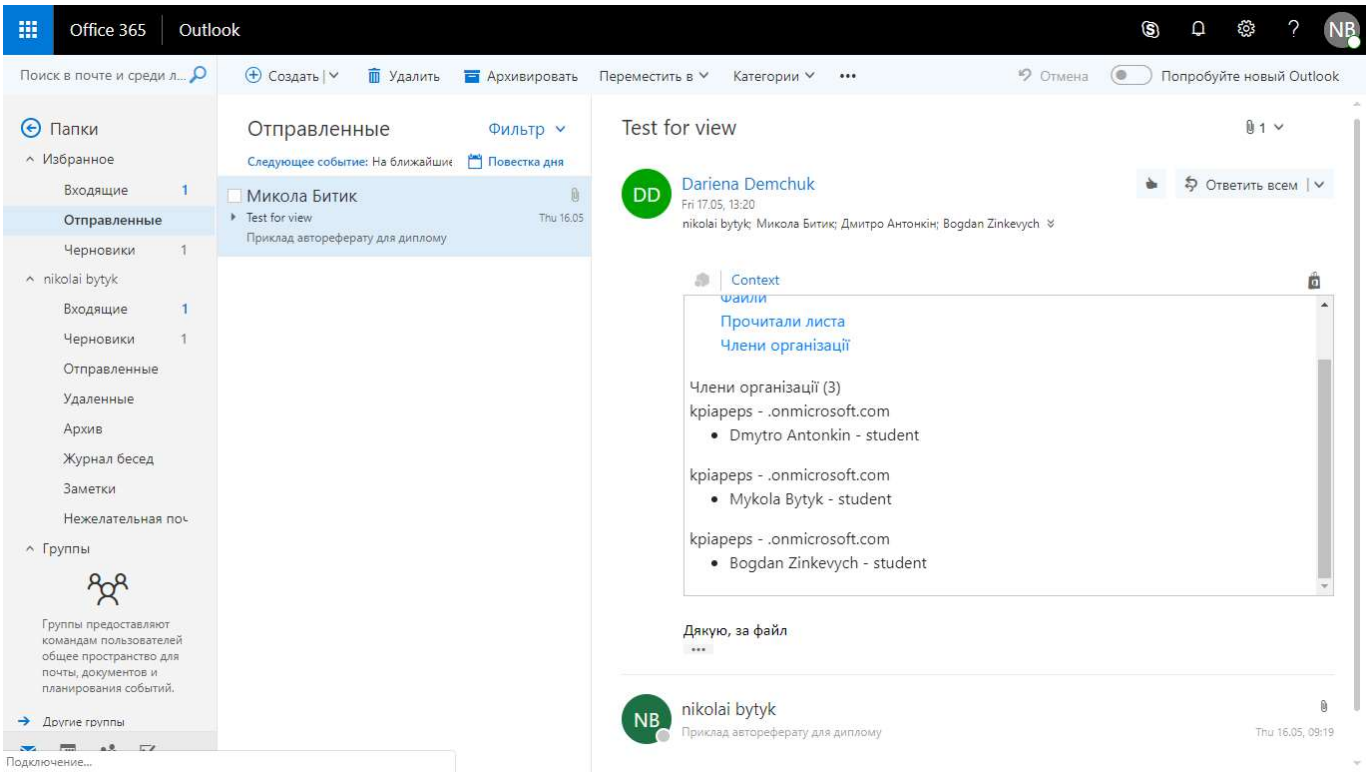

Рисунок 5.12 – інформація про членів масової розсилки

Застосунок було інтегровано в інтерфейс Outlook, що дозволяє працювати з електронною поштою в звичному режимі, а при необхідності мати змогу використовувати застосунок.

ВИСНОВКИ

В процесі аналізу підприємств було виявлено, що існує наявна проблема з комунікацією між членами організацій, в зокрема в роботі з наказами та дорученнями. Кожна організація потребує своїх індивідуальних функцій від системи. Що не дає змогу великим корпораціям зробити один універсальний застосунок для задоволення потреб організацій.

Враховуючи вищезазначені проблеми було вирішено, що необхідно розробити застосунок, який буде розширювати стандартний функціонал платформи Microsoft Office 365.

Реалізовані наступні завдання:

— проведений порівняльний аналіз існуючих варіантів впровадження до платформи Microsoft Office 365

— створення програмного застосунку з інтуїтивно інтегрованим в Outlook інтерфейсом

— розробив архітектуру програмного застосунку для вилучення необхідної інформації, що зберігається на платформи office 365

— об’єднані існуючий функціонал та розроблений новий

— розроблений алгоритм оцінювання важливості навиків

Такі дії допожуть вирішити проблему контролю за підлеглими.

Потенційними користувачами програмного продукту є керівники підприємств та навчальних закладів, члени організацій.

При розробці додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365 я використав середовища розробки JetBrains WebStorm 2017, веб-інтерфейсу та серверу Node.js, мову програмування JavaScript та мову розмітки HTML, фреймворк AngularJS.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Office 365 Business преміум: [електронний ресурс] Режим доступу: https://products.office.com/uk-ua/business/office-365-business-premium (дата звернення: 12.04.2019).

2. ПЗ для спільної роботи та групового чату – Microsoft Teams: [електронний ресурс] Режим доступу: https://products.office.com/uk-ua/ microsoft-teams/group-chat-software (дата звернення: 12.04.2019).

3. Microsoft Office 365 [Електроний ресурс] //Вікіпедія: [електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/ Microsoft_Office_365 (дата звернення: 12.04.2019).

4. Соммервилл И. Инженерия программного обеспечения. «Вильямс», 2002. 624 р.

6. Майкл Ноэл К.С. Microsoft SharePoint 2010. Полное руководство. Вильямс, 2012. 880 р

7. Withee R. SharePoint 2016 For Dummies. For Dummies, 384.

8. WebStorm: [електронний ресурс] Режим доступу: https://ru.wikipedia.org/ wiki/WebStorm (дата звернення: 08.11.2018).

9. JSON: [електронний ресурс] [2018]. Режим доступу: https:// uk.wikipedia.org/wiki/JSON

10. Introducing JSON: [електронний ресурс] [2018]. Режим доступу: https:// www.json.org/

11. Kocherhin O. Mastering Vue.js. 2018. 362 p.

12. Руби С. Т.Д..Х.Д.Х. Гибкая разработка веб-приложений в среде Rails. 4-е изд. СПБ: Питер, 2012. 464 с.

13. Maniatis K., Kyriakidis A., та You. The Majesty of Vue.js 2. 2017. 333 p.

14. SSR. Отрисовкана стороне сервера: [електронний ресурс] [2018]. Режим доступу: https://ru.vuejs.org/v2/guide/ssr.html#Nuxt-js (дата звернення: 12.04.2018).

15. Gore A. Full-Stack Vue.js 2 and Laravel 5. Packt Publishing, 2017. 378 p.

16. Руководство по серверному рендерингу Vue.js 2018. URL: https://ssr.vuejs.org/ru/#нужен-ли-вам-ssr.

17. Файн Я., Моисеев А. Angular и TypeScript. Сайтостроение для профессионалов. Питер, 2018. 464 р.

18. METANIT - Что такое TypeScript: [електронний ресурс] [2018]. Режим доступу: https://metanit.com/web/typescript/1.1.php (дата звернення: 9.11.2018).

19. JavaScript. Библия пользователя = JavaScript. Bible / Денни Гудман (Danny Goodman), Майкл Моррисон (Michael Morrison); пер. с англ. И. В. Василенко. — 5-е

20. Alexei White. Major JavaScript Engines // JavaScript Programmer's Reference. — Indianapolis, IN 46256: Wiley Publishing, Inc., 2009. — P. 12 — 13.

21. Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. AJAX в действии: технология — Asynchronous JavaScript and XML = Ajax in Action. — М.: Вильямс, 2006. — С. 640. — ISBN 1-932394-61-3.

22. JavaScript. Карманный справочник. Необходимый код и команды = JavaScript. Phrasebook. Essential code and commands / Кристиан Уэнц (Cristian Wenz); пер. с англ. И. В. Берштейн. — Москва, Санкт-Петербург, Киев: ООО "И. Д. Вильямс", 2008. — С. 18. — 272 с.

23. Офіційна сторінка NodeJS на російській мові [Електронний ресурс]. — Режим доступа: http://nodejs.ru/ — (дата звернення: 02.05.2019).

24. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и Java RMI. – М.: Мир, 2002. –510 с.

25. Боггс У., Боггс М. UML Rational Rose. Бестселлер #, Лори.– Москва, 2000.- 580 с.

ДОДАТОК 1

Розробка додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365.

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ51100_19Б

Аркушів 3

Київ 2019

```
'use strict';

var fs = require('fs'),
    express = require('express'),
    http = require('http'),
    https = require('https');

var https_options = {
  key: fs.readFileSync('./hostkey.pem'),
  cert: fs.readFileSync('./hostcert.pem')
};

var PORT = 8443,
    HOST = 'localhost';

var app = express();

// set static routes
app.use('/', express.static(__dirname + '/src'));
app.use('/vendor', express.static(__dirname + '/bower_components'));
app.use('/template',                express.static(__dirname                +
'/bower_components/ui.bootstrap/template'));

//https://localhost:8443/ui.bootstrap/template/accordion/accordion-group.html

var server = https.createServer(https_options, app)
                  .listen(PORT, HOST);


var httpServer = http.createServer(app);
var httpOptions = {
port: 80,
host: 'localhost'
};

httpServer.listen(httpOptions);

console.log('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+');
console.log('HTTPS Server listening @ https://%s:%s', HOST, PORT);
console.log('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+');
```

```
var appConf = {
  tenantName: "kpiapeps"
};

var routeConf = {};
    routeConf['tenant']         = appConf.tenantName + '.onmicrosoft.com';
    routeConf['clientId']       = 'f82fd4d0-317c-4230-ac2a-a5101ab77ff0';
    routeConf['cacheLocation']  = 'localStorage';
    routeConf["endpoints"] = {};
    routeConf.endpoints['https://'        +        appConf.tenantName        +
'.sharepoint.com/_api/'] = 'https://' + appConf.tenantName + '.sharepoint.com';
    routeConf.endpoints['https://'      +      appConf.tenantName      +      '-
my.sharepoint.com/_api/v1.0/me']    =    'https://'    +    appConf.tenantName    +    '-
my.sharepoint.com';
    routeConf.endpoints['https://outlook.office365.com/api/v1.0/me']          =
'https://outlook.office365.com';

(function () {
  'use strict';

  var outlookApp = angular.module('appowa');

  // load routes
  outlookApp.config(['$routeProvider',                          '$httpProvider',
'adalAuthenticationServiceProvider', routeConfigurator]);

  function routeConfigurator($routeProvider, $httpProvider, adalProvider) {

    //Initialize ADAL
    adalProvider.init(routeConf, $httpProvider);

    $routeProvider
        .when('/', {
          templateUrl: '/views/home-view.html',
          controller: 'homeController',
          requireADLogin: true
        })
        .when('/files', {
          templateUrl: '/views/files-view.html',
          controller: 'homeController',
          controllerAs: 'vm',
          requireADLogin: true
        })
```

```
        .when('/mails', {
          templateUrl: '/views/mails-view.html',
          requireADLogin: true
        })
        .when('/employees', {
          templateUrl: '/views/employees-view.html',
          requireADLogin: true
        })
        .when('/reports', {
          templateUrl: '/views/reports-view.html',
          requireADLogin: true
        });
    $routeProvider.otherwise({redirectTo: '/'});
  }
})();
```

ДОДАТОК 2

Розробка додатку для управління інформаційними потоками електронної пошти outlook на базі платформи office 365.

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ51100_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ51100_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ51100_19Б	server.js app.routes.js	Основні компоненти